Chapter 4: Configuring the Shell

Shell Variables

- A name that holds a value
- Useful to store key system information as well as shell data
- Can also be used to modify how specific commands behave
- Examples:

  NAME=Bob

  LPDEST=hplaserjet

Shell Variables

- Variable names must begin with alpha or underscore characters
- Remaining characters in variable name can be alphanumeric and underscore characters

Local vs Environment Variable

- Local variables
  - Only available to shell they are created in
  - By convention, names are all lower case
  - Can be exported for use by subshells
  - Format: var=*value*
- Environment variables
  - Passed into commands opened by the shell
  - There are predefined environmental variables (PATH, TZ, etc.)
  - Format: var=*value*; export var

Ways to Make a Local Variable and Environmental Variable

- Environment variables can also be set the following ways

```
$export var=value
$declare -x var=value
$typeset -x var=value
```

Displaying Variables

- To display all variables, use the `set` command
- To display only environment variables, use one of the following commands:

```
$env
$declare -x
$typeset -x
$export -p
```

- To display a variable:

```
$echo $var
```
(i.e. $echo $PATH)

Setting/Unsetting Variables Summary

- To set a local variable:

```
$var=value
```

- To set an environment variable, run two commands:

```
$var=value
$export var
 Or
$var=value; export var
```

- To unset a variable:

```
$unset var
```

The PATH Variables

- Used to search for commands that are entered by the user so they can be executed
- Format: dir1:dir2:dir3
- Searches in order, dir1 first, dir2 second, etc.
- To set:

  `PATH=$PATH:`*`/new/path`*

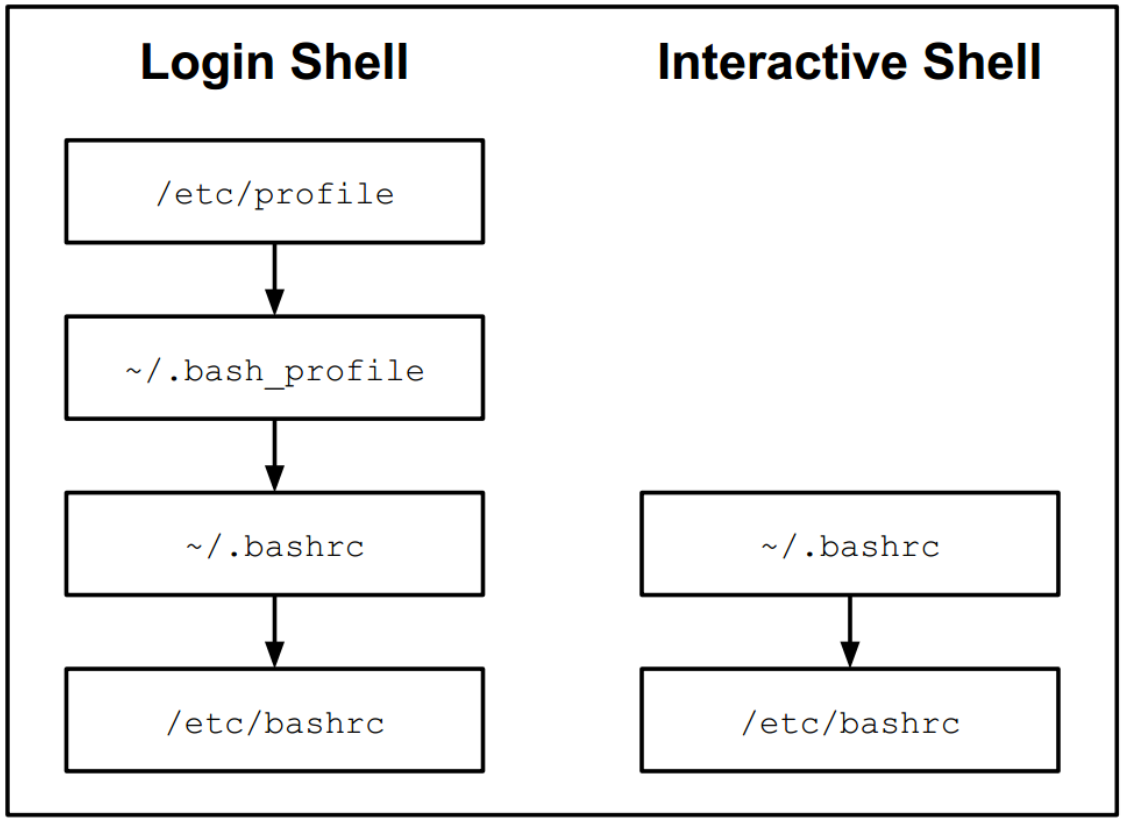
Initialization Files

- Contain a series of commands, settings, and variables that establish the working environment and preferences. For example:
  - -Default prompt
  - -Default printer
  - -Default list of directories to search for commands
- Two types:
  - - Global – affect all system users; located in /etc
  - - Local – user specific; located in a user's home directory and read after the global init files

Bash Startup

- The bash shell can be started in two different ways:
  - Login Shell: When a shell is provided to the user during login
  - Interactive Shell: When the kernel automatically starts a new shell to run a program or when a user starts a new shell manually. Also called an Non-login Shell

Initialization File Summary



Which File Should You Edit?

File	Purpose
`/etc/profile`	This file can only be modified by the administrator and will be executed by every user who logs in. Administrators use this file to create key environment variables, display messages to users as they log in and set key system values.
`~/.bash_profile`	Each user has their own `.bash_profile` file in their home directory. The purpose for this file is the same as the `/etc/profile` file, but having this file allows a user to customize the shell to their own tastes. Normally used to create customized environment variables.
`~./bashrc`	Each user has their own `.bashrc` file in their home directory. The purpose for this file is to generate things that need to be created for each shell, such as local variables and aliases.
`/etc/bashrc`	This file may affect every user on the system. Only the administrator can modify this file. Like the `.bashrc` file, the purpose for this file is to generate things that need to be created for each shell, such as local variables and aliases.

Bash Exit Scripts

- When bash exists, it executes the following files:

```
~/.bash_logout
/etc/bash_logout
```

- Useful to place commands to "clean up" your account, like to delete old files and clear screen

Command History

- Bash stores previous commands in memory
- You can re-execute these commands quickly
- When you log out, the commands are stored into the `~/.bash_history` file
- When bash starts, these commands are read back into memory

Executing Previous Commands (method 1)

Action	Key	Alternate Key Combination
Previous history item	Up arrow	CTRL+p
Next history item	Down arrow	CTRL+n
Reverse history search		CTRL+r
Beginning of line	Home	CTRL+a
End of line	End	CTRL+e
Delete current character	Delete	CTRL+d
Delete to left of cursor	Backspace	CTRL+x
Move cursor left	Left arrow	CTRL+b
Move cursor right	Right arrow	CTRL+f

Changing History Editing Keys

- History commands use emacs editor commands by default.
- Can change to vi editor commands by executing:

```
$set -o vi
```

- To make this change automatic, place the following in the `~./inputrc` file:

```
set editing-mode vi
set keymap vi
```

Using the history Command

- Displays a list of previously executed commands:

```
$ history
1 ls
2 cd test
3 ls -l
4 history
```

- Common history command options:

Option	Purpose
`-c`	Clears the list
`-r`	Read the history file and replace the current history
`-w`	Write to the current history list to the history file

Configuring the history Command

- The HISTFILESIZE variable indicates how many commands to store in the history file:
  - HISTFILESIZE=500
- The HISTSIZE variable indicates how many commands to store in memory:
  - HISTSIZE=100
- The HISTIGNORE variable can be used to tell bash to not store certain commands in the history list:
  - HISTIGNORE='ls*:cd*:history*:exit'

The HISTCONTROL variable

- The HISTCONTROL variable changes what is stored in history:
  - HISTCONTROL=ignoredups will prevent duplicate commands that are executed consecutively
  - HISTCONTROL=ignorespace will not store any command that begins with a space
  - HISTCONTROL=ignoreboth will not store consecutive duplicates or any command that begins with a space

Executing Previous Commands (method 2)

- Alternative to using navigation (i.e. ↑,↓) or Ctrl + key combinations (i.e. CTRL+p)
- The "!" is a special character to the bash shell to indicate the execution of a command within the history list:

History Command	Meaning
!!	Repeat the last command
!-4	Execute the command that was run four commands ago
!555	Execute command number 555
!ec	Execute the last command that started with "ec"
!?grep	Execute the last command that contained "grep"